METHOD AND PROGRAMMABLE DEVICE FOR TRIANGLE INTERPOLATION IN HOMOGENEOUS SPACE

REFERENCE TO EARLIER APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application Serial No. 60/419,881, filed October 19, 2002, and entitled "METHOD AND PROGRAMMABLE DEVICE FOR TRIANGLE INTERPOLATION IN HOMOGENEOUS SPACE," which application is incorporated herein by reference.

FIELD OF THE INVENTION

[0002] The present invention relates generally to graphics processing, and more particularly to triangle interpolation processing in homogeneous space.

DESCRIPTION OF THE RELATED ART

[0003] Polygon (triangle) interpolation is one of the computationally intensive tasks in graphics hardware because this interpolation is done for each pixel of each object in a scene. Each triangle may have several attributes like depth, colors, numerous textures and the quality of the image generated is greatly dependent on the accuracy of interpolation. The overall frame generation rate also depends on interpolation speed. The hardware implementation of these functions takes significant part of the gate budget of a modern graphics chip and is one the most critical parts of the chip.

[0004] Several solutions to the triangle interpolation problem have been implemented in different graphics architectures. Most of the current approaches fall into the following three categories: (i) fixed point interpolation in screen Cartesian coordinates with perspective correction; (ii) fixed point interpolation in screen barycentric coordinates with perspective correction; and (iii) fixed point interpolation in homogeneous coordinates.

[0005] Approaches in the first two categories require projecting of all parameters to screen space (i.e., division by W) for further interpolation and later perspective correction (i.e., division by 1/W) per pixel. These approaches are shown in FIGs. 1 and 2, respectively.

[0006] The third approach avoids the redundant projection of parameters to screen space with further correction and calculates the same homogeneous barycentrics, as shown in FIG. 3.

Each of the above approaches performs calculations on a per pixel basis and in case of medium and large triangles requires a substantial number of calculations. Also, approaches in the three categories use significant amount of dedicated hardware for implementation, hardware which cannot be shared or used for other computing tasks.

[0007] More particularly, disadvantages of the first approach, fixed point interpolation in screen Cartesian coordinates with perspective correction, include: (a) redundant calculation with projection (division by W) of all parameters to screen space to make it linear; (b) the steps of project, interpolate, and correct to recover the true value; (c) redundant parameter delta setup calculation for interpolation including 1/W (accuracy problems arise); (d) redundant true parameter value recovery in each pixel by dividing to interpolated 1/W_pix value; and (e) a significant amount of dedicated hardware.

[0008] Disadvantages of the second approach, fixed point interpolation in screen barycentric coordinates with perspective correction, include: (a) same redundant calculation with projection (division by W) of all parameters to screen space to make it linear; (b) the steps of project, interpolate, and correct to recover the true value; (c) same redundant true parameter value recovery in each pixel by dividing to interpolated 1/W_pix value; and (d) a significant amount of dedicated hardware, which cannot be used for any other tasks.

[0009] Disadvantages of the third approach, fixed point interpolation in homogeneous coordinates, include: (a) calculations for pixel barycentrics must all have been done at the pixel level and, in the case of multi-pixel triangles, the number of calculations grows multiplicatively; and (b) a significant amount of hardware dedicated to the task..

[00010] Thus, there is a need for a method and apparatus pertaining to polygon interpolation in graphics hardware that significantly reduces the number of calculations needed and does not require a significant amount of dedicated hardware.

BRIEF SUMMARY OF THE INVENTION

[00011] The present invention is directed towards the above-mentioned need. The present invention addresses the problem of fast, accurate and efficient polygon interpolation in graphics hardware to find correct value of each parameter (Z, color, multiple texture coordinates) of every pixel of triangle. For algorithm implementation, the present invention uses a programmable single

instruction multiple data (SIMD) scalar unit, which can be also used for further pixel processing according to Microsoft Dx9,10 and OpenGL API requirements for programmable graphics machines. The triangle interpolation algorithm of the present invention can be executed on this programmable SIMD scalar unit with efficiency that is close to the efficiency of dedicated hardware. In fact, all interpolation operations are executed in floating point arithmetic with the highest possible accuracy.

[00012] In accordance with a purpose of the invention as described herein, a method is provided for obtaining an attribute in homogenous space. The method steps include obtaining the vertices of a triangle, each vertex being represented by a set of coordinates in a world coordinate space and having an attribute. For each vertex, the method further includes transforming the world space coordinates and the attribute of the vertex to coordinates and an attribute in viewer space, computing a set of homogenous coefficients of the vertex based on the viewer space coordinates, and projecting the viewer space coordinates of the vertex to coordinates in screen space. The method also includes determining, in the screen space, pixels that are affected by the triangle based on the screen space coordinates. Lastly, for each pixel affected by the triangle, the method additionally includes computing, based on the homogenous coefficients, a set of barycentric coefficients in homogenous space, and performing a linear interpolation based on the set of homogenous barycentric coefficients and the attributes of the vertices in the viewer space to obtain the attribute in the homogenous space of that pixel.

[00013] One advantage of the present invention is the invention uses fewer arithmetic operations compared to prior art implementations and, thus, has better performance compared to triangle interpolation in screen space and later perspective correction.

[00014] Another advantage of the present invention is that it avoids parameter projection for interpolation in screen space and further perspective correction computations to recover true value in the viewpoint space.

BRIEF DESCRIPTION OF THE DRAWINGS

[00015] These and other features, aspects and advantages of the present invention will become better understood with regard to the following description, appended claims, and accompanying drawings where:

FIG. 1 shows a prior art triangle rasterization and interpolation method using screen Cartesian coordinates;

FIG. 2 shows a prior art triangle rasterization interpolation method using screen barycentric coordinates;

FIG. 3 shows another prior art triangle interpolation method using homogeneous barycentric coordinates;

FIG. 4A shows a flow chart of an interpolation method in accordance with the present invention;

FIG. 4B illustrates the triangle in a) world coordinate space b) homogenous coordinate space and c) screen coordinate space with pixels affected by the triangle;

FIG. 4C & D together show the triangle setup and pixel processing formulae in accordance with the present invention;

FIG. 5A shows a comparison of the steps of FIG. 4 with FIG. 3;

FIG. 5B shows a table that tabulates the operations of FIG. 3 and FIG. 4;

FIG. 6A shows a programmable unit for triangle barycentric interpolation in long data processing mode;

FIG. 6B shows an instruction set for the programmable unit; and

FIG. 7 shows an ALU structure for the short data processing mode of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[00016] In a graphics system embodying the present invention, primitives are decomposed into triangular polygons that are further processed to provide a better representation of the images. To illustrate such processing, FIG. 4A is a flow chart of a triangle interpolation method in accordance with the present invention. Specifically, this method is for triangle rasterizing and linear parameter interpolation using barycentric coordinates in a homogenous space. According to this method, two levels of coefficient calculations take place, calculations at the triangle level and calculations at the pixel level, where certain operations occur concurrently as will be further explained below. It is noted

that the steps are shown in the oval-shaped boxes and the resulting data sets are shown in the rectangular-shaped boxes.

[00017] To allow better understanding of the method in FIG. 4A, the illustrations in FIG. 4B include the triangle in a) world coordinate space 120 b) homogenous coordinate space 120' and c) screen coordinate space with pixels affected by the triangle 120". In the world coordinate space, the triangle 120 is defined by three vertices, P1, P2, P3, with respective coordinates [X1, Y1, Z1], [X2, Y2, Z2], [X3, Y3, Z3], and respective attributes $p1a$, $p1b$, ..., $p2a$, $p2b$, ..., $p3a$, $p3b$, ...

[00018] Thus, in the triangle level, the triangle data set is first characterized in the world coordinate space by the three vertices with texture coordinates as follows:

$$\{[X1, Y1, Z1,1][p1a, p1b,...]\}, \{[X2, Y2, Z2,1][p2a, p2b,...]\}, \{[X3, Y3, Z3,1][p3a, p3b,...]\}$$

[00019] The world coordinate space (three-dimensional space) serves to define objects prior to any geometric transformations. Given this triangle data set in the world coordinate space, the triangle data set is then transformed into viewer coordinate space and clipped to a view volume. Essentially, the triangle first specified in world space is treated with respect to a defined viewpoint. In the viewer space, the eye or viewpoint is the origin for coordinates and the view ray is along the Z-axis. The resulting triangle data set in viewer coordinate space, i.e., the triangle 120' with coordinates in viewer (homogenous) space as shown in Fig. 4B part (b), is described as follows:

$$\{[X1hY1h, Z1h, W1][p1ah, p1bh,...]\}, \{[X2h, Y2h, Z2h, W2][p2ah, p2bh,...]\},$$
$$\{[X3h, Y3h, Z3h, W3][p3ah, p3bh,...]\},$$

where [X1h, Y1h, Z1h], [X2h, Y2h, Z2h], [X3h, Y3h, Z3h], are the respective coordinates in the homogenous space, W1, W2, W3 are the perspective correction parameters, and $p1ah$, $p1bh$, ..., $p2ah$, $p2bh$, ..., $p3ah$, $p3bh$, ...are the attributes in the homogenous space.

[00020] A triangle setup phase follows the transformation to viewer coordinate space, in which, three (3) sets of the homogenous coefficients (ai, bi, ci) are derived from the triangle data set in the homogenous coordinate space, where, for example, a1=Y2h*W3 - Y3h*W2. More broadly, the coefficients are derived by:

$$a_i = Yjh \cdot Wk - Ykh \cdot Wj;$$

$$b_i = Xjh \cdot Wk - Xkh \cdot Wj; \text{ and}$$

$$c_i = Xjh \cdot Ykh - Xkh \cdot Yjh,$$

and where i, j & k = 1, 2 & 3.

[00021] Concurrently, as is preferred, the triangle viewer coordinate data set is projected to the screen coordinate space, to produce {[X1s, Y1s], [X2s, Y2s],... [X3s, Y3s]}.

[00022] Screen coordinate space involves further transformations where the object geometry in screen coordinate system is projected and transformed to give depth to the displayed object. Accordingly, as shown in FIG. 4A, the triangle data set is then rasterized to produce the blank pixel data. Rasterization dissects primitives such as triangles into pixels and in the process it performs Z-buffering (to capture depth, Z-axis, data) and other pixel-level functions. Thus, for N pixels covered by the triangle (120" as shown in FIG. 4B, part (c)), the blank pixel data from the rasterization step is described by the N coordinates in the screen space as follows:

{[X1s, Y1s], [X2s, Y2s],... [XNs, YNs]}.

[00023] As further shown in FIG. 4B, a pixel setup phase follows. Namely, for each pixel, P, covered by the triangle, homogenous space coefficients dl, d2, d3 and barycentric coefficients, α, ß, and γ, are calculated in the homogeneous space. From this we arrive at the homogeneous barycentric data for the pixels affected by the triangle. The pixel setup phase uses the data from the triangle level setup phase and the blank pixel data from the rasterization step.

[00024] Lastly, in the interpolation phase, the homogeneous coordinates and attributes for each pixel are linearly interpolated using the barycentric coordinates previously calculated. This includes linear interpolation of the Z-coordinate in homogenous space, Zh, the perspective correction parameter, W, and the attributes, Pah, Pbh,... in the homogenous space. For each pixel, the resulting Z-coordinate , Z_pix, is:

$$Z\_pix = \alpha \cdot Z1h + \beta \cdot Z2h + \gamma \cdot Z3h;$$

The resulting perspective correction parameter, W_pix, is:

$$W\_pix = \alpha \cdot W1 + \beta \cdot W2 + \gamma \cdot W3;$$ and

for each of its attributes, after interpolation the resulting attributes, Phi, for each pixel, is

$$Phi\_pix = \alpha \cdot P1hi + \beta \cdot P2hi + \gamma \cdot P3hi.$$

After the linear interpolation, the resulting pixels data with depth is described as:

$$\{[X1s, Y1s, Z1s, W1h][p1a, p1b, ...]\}, \{[X2s, Y2s, Z2s, W2h][p2a, p2b, ...]\},$$
$$\{[X3s, Y3s, Z3h, W3h][p3a, p3b, ...]\}, ...$$

[00025] More particularly, together, FIGs. 4C & 4D show triangle setup and pixel processing formulae in the various steps accordance with the present invention. As shown, for each pixel of the triangle, we can express the pixel's coordinates from three vertices using the barycentric coefficients, as typically employed in texture application techniques. A barycentric coordinate system is defined, for example, in OpenGL™ from Silicon Graphics, Inc. Given a triangle with three vertices, P1 (x1, y1, z1), P2(x2, y2, z2), and P3 (x3, y3, z3), a point P (x, y, z) located somewhere in the plane of the triangle may be defined in terms of the three vertices using $P = \alpha P1 + \beta P2 + \gamma P3$, where α, β, and γ are the barycentric coefficients of the point P. Consequently, the point P can be defined as follows:

$X = \alpha x1 + \beta x2 + \gamma x3$, and

$Y = \alpha y1 + \beta y2 + \gamma y3$.

[00026] It is noted that the point P is defined by a unique set of barycentric coordinates and further that the coordinates satisfy the requirement that $\alpha + \beta + \gamma = 1$.

[00027] Thus, based on the barycentric coefficients, the coordinates, X,Y, of each pixel affected by the triangle can be derived. This means that the point P can be calculated as a linear combination of the vertices (P1, P2, P3). As the barycentric coefficients are applicable to interpolation of any attribute of the pixel, including Z-coordinate, color, UV coordinates, etc., once they are calculated the attributes can be calculated as well. In particular, α, β, and γ are calculated as shown in Equation (1) of FIG. 4C. And, once a determination of the barycentric coefficients is made they can be used to interpolate the entire polygon (triangle). Furthermore, with the ability to locate each pixel point (P) using the barycentric coordinates in the homogenous space, the polygon can be shaded or texture mapped by specifying the locations of the points.

[00028] When the three triangle vertices are enumerated with i=1, 2, 3, j= i mod3+1, and k= j mode3+1, and the substitutions into Equation (1) of $a_i=y_i-y_k$, $b_i=x_k-x_j$, $c_i=x_j*y_k - x_k*y_j$, are made, as shown in Equation (2), the barycentric coefficients can be restated as shown in Equation (3).

[00029] As shown in Equation (4), w, the perspective correction parameter of the point P, is a function of the barycentric coordinates, α, β, and γ, and the perspective correction parameters, w_1, w_2, w_3, of the three vertices. It is noted that the perspective correction parameter, w, is preserved in order to

preserve the information on the distance to the viewpoint. However, before re-calculation of the barycentric coordinates in the homogenous (viewpoint) space can be done, the vertices, x_i, y_i, are converted to coordinates in the homogenous space, $\tilde{x}_i, \tilde{y}_i$, using the corresponding parameters, w_i. The coordinates $(\tilde{x}, \tilde{y})$ of P in the homogenous space are then derived using w.

[00030] Then, a number of calculations are performed to allow for the calculation of the barycentric coordinates in the homogenous space. These include 1) calculating three sets of homogenous space coefficients, a_i, b_i, c_i, (per triangle) which are shared for every pixel affected by the triangle, 2) converting these to coefficients in the homogenous space, $\tilde{a}_i, \tilde{b}_i, \tilde{c}_i$, using the perspective correction parameters, w_i, and 3) calculating d_i (per pixel), as shown in Equation (6) of FIG. 4D. Once the three values, d_1, d_2, d_3, are determined, the barycentric coefficients (for P) in the homogenous space, $\tilde{\alpha}$, $\tilde{\beta}$, and $\tilde{\gamma}$, can be calculated (for P) to complete the triangle interpolation in the homogenous space, as shown in Equation (5). Through the interpolation process for each point, P, of a pixel affected by the triangle, the barycentric coefficients calculated in the homogenous space are used to obtain the true value (avoiding a division by 1/W). Thus, the triangular object is simulated by pixels with proper depth attributes.

[00031] FIG. 5A shows a comparison of the steps of FIG. 4A with FIG. 3. The shaded blocks on the left side of the diagram indicate steps in FIG. 3 that are different and the shaded blocks on the right side of the diagrams indicate steps in FIG. 4A that are different. Unshaded blocks in FIG. 5A are steps common to both FIGs. 3 and 4A. FIG. 5B shows a table that tabulates the differences in the number of operations between the steps of FIG. 4A and FIG. 3.

[00032] The difference between the approach of the present invention and the method of FIG. 3 is that, in the present invention, the homogeneous barycentric calculation is split into two levels: (a) a triangle level setup and (b) a pixel level setup. This reduces the number of calculations for multi-pixel triangles. Another difference is that the present invention uses fewer arithmetic operations compared to prior art implementations and, thus, has better performance compared to triangle interpolation in screen space and later perspective correction. Additionally, the method of the present invention avoids parameter projection for interpolation in screen space and further perspective correction computations to recover true value in the viewpoint space.

[00033] Yet another difference is that the present invention provides very accurate and fast interpolation of triangles using the same ALUs for both triangle and pixel processing, with arbitrary interleaving of triangle and pixel processing instructions. The hardware SIMD unit used for interpolation is scalable to increase the performance of barycentric interpolation. The hardware unit is very flexible in terms of accuracy and performance, and it can be used for other graphics processing tasks to improve overall performance balance. Indeed, the programmable SIMD scalar unit, used for the algorithm implementation, can be used for further pixel processing according to Microsoft Dx9,10 and OpenGL API requirements for programmable graphics machines. Also, the triangle interpolation method, in accordance with the present invention, is more suitable for implementation on programmable graphics processing unit and gives an efficiency close to the efficiency of a dedicated hardware implementation.

[00034] An apparatus of the present invention is implemented in one universal programmable unit replacing separate depth, color and texture address interpolators. To illustrate, FIG. 6A is a diagram of a programmable hardware unit 10 in accordance with an embodiment of the present invention. The universal hardware unit 10 includes a programmable single instruction multiple data (SIMD) scalar unit 12 for executing the various interpolation operations. Notably, the triangle interpolation algorithm of the present invention can be executed on the programmable SIMD unit 12 with efficiency that is close to the efficiency of dedicated hardware. In fact, all interpolation operations are executed in floating point arithmetic with the highest possible accuracy. To increase the performance of the hardware unit 10, the SIMD 12 can be replicated.

[00035] This hardware unit 10 is implemented as a two-ALUs unit where the scalar ALUs, 16 and 18, are implemented with shifted processing cycles. The cycle shifting is accomplished with the bypass registers 30 and 32. The unit 10 can process, in each clock, one triangle or a number of pixels (e.g., 2-4 pixels) depending on the required accuracy, and with the arbitrary interleaving of triangle and pixel processing instructions. The Reciprocal unit 20, is used for processing the division calculations (as described with reference to the method of FIG. 4A and FIGs. 4C & 4D). The SIMD Processing algorithm is programmed in microcode memory 14 and the instruction set is optimized to reduce the number of instructions.

[00036] Also included in the hardware unit 10, are the triangle rasterizer 22, and the blank pixel screen coordinates data it produces are stored in pixel memory 24. The vertex

geometry processing unit 26, produces data of the triangle coordinates in the homogenous space and this data is stored in the triangle memory unit 28.

[00037] FIG. 6B sets forth the SIMD ALU instruction set. Two types of data, long floating point (FP) and short FP, can be processed in separate or mixed mode for reducing the length of the program. One type of the instruction used to perform the required calculations in Equation (6) in FIG. 4D is the 'Cross Mode Long' instruction, XPRDL D,D", P0, P1 for producing a cross product. For example, the instruction "XPRDL A0, V1.yw, V2.yw" is executed to calculate the homogenous coefficient a1. Then, for calculating the value di, one of the instructions is the 'Folded Long' instruction, FSUBL D, P0", P2" which produces cross products.

[00038] To see how these and other instructions are used, an example program is set forth below. In this program, the instruction XPRDL is used a number of times to perform the calculations required by Equation (6) for i=0, 1, and 2 (i.e., for a0, a1, a2, b0, b1, b2, c0, c1, c2). Note that the index 'i' in this program equals 0,1,2 rather than i=1,2,3, as described above, but the results are not affected by this. For pixel barycentric coefficients calculations (α,β,γ), as required by Equation (5), the instructions include MOV, FBL ('Folded Blend Mode'), FWD, FSUBL ("Folded Long'), etc. For pixel attribute interpolation as shown in FIG. 4A, the instructions include MULL (multiply long) and FSUBL.

[00039] More specifically, below is the listing of the required calculations and their implementation in the instruction set of the SIMD ALU, for the (i) triangle setup and (ii) pixel processing, and (iii) pixel attribute interpolation.

```
//Barycentric triangle setup:
//The calculations as required by Equation (6) for index i=0:
# tri_a[0] = v[1].y * v[2].w - v[2].y * v[1].w
# tri_b[0] = v[2].x * v[1].w - v[1].x * v[2].w
# tri_c[0] = v[1].x * v[2].y - v[2].x * v[1].y
//
//The calculations as required by Equation (6) for index i=1:
# tri_a[1] = v[2].y * v[0].w - v[0].y * v[2].w
```

tri_b[1] = v[0].x * v[2].w - v[2].x * v[0].w

tri_c[1] = v[2].x * v[0].y - v[0].x * v[2].y

//

//The calculations as required by Equation (6) for index i=2:

tri_a[2] = v[0].y * v[1].w - v[1].y* v[0].w

tri_b[2] = v[1].x * v[0].w- v[0].x * v[1].w

tri_c[2] = v[0].x * v[1].y - v[1].x * v[0].y

[00040] To implement the forgoing required calculations, the following instructions are executed (with 9 instructions, for SIMD 4 bits (nibble) mode):

//

//The instructions for executing calculations as required by Equation (6) for index i=0:

XPRDL A0, V1.yw, V2.yw
XPRDL B0, -Vl.xw, V2.xw
XPRDL C0, V1.xy, V2.xy
//

//The instructions for executing calculations as required by Equation (6) for index i=1:

XPRDL Al, V2.yw, V0.yw
XPRDL Bl, -V2.xw, V0.xw
XPRDL C1, V2.xy, V0.xy
//

//The instructions for executing calculations as required by Equation (6) for index i=2:

XPRDL A2, V0.yw, V1.yw
XPRDL B2, -V0.xw, V1.xw
XPRDL C2, V0.xy, V1.xy

//Pixel barycentric data processing, as required by Equation (5):

//Barycentric parameters calculation for each pixel, with calculations of di for i=0,1,2:

pix_d[0] = x * tri_a[0] + y * tri_b[0] + tri_c[0]

pix_d[1] = x* tri_a[1] + y* tri_b[1]+tri_c[1]

pix_d[2] = x * tri_a[2] + y * tri_b[2] + tri_c[2]

pix_d[3] = 1 / (pix_d[0] + pix_d[1] + pix d[2])

//

//Pixel barycentric coefficients (α,β,γ)calculation based on di:

alpha = pix_d[0] * pix_d[3]

#beta = pix_d[1] * pix_d[3]

gamma = 1 - alpha - beta

[00041] To implement the forgoing required calculations, the following instructions are executed (with 7 instructions):

//

```
MOV pr, xy
FBLDL D0, pr, AB0, C0
FBLMP NULL, D1, pr, AB1, C1, AB2, C2
FWD RCPL
MULL ALPHA, D0, ACC
MULL BETA, D1, ACC
FSUBL GAMMA, HW_ONE, ALPHA, 0, BETA
```

//Pixel attribute interpolation:

//Required calculations:

pix_att = v_att0 * alpha + v_attl * beta + v_att2 * gamma

Z interpolation mode - long operands

[00042] To implement the forgoing required calculations, the following instructions are executed (with 2 instructions, for SIMD 8 bits (byte) mode):

//

MULL R0, ALPHA, V_ATT0
FBLDL PIX_ATT, BETA, V_ATT1, R0, GAMMA, V_ATT2

//Other attribute mode - short parameter operands (with 3 instructions, for SIMD 16 bits (word) mode):

//
MULM r0, v_att0, ALPHA
MADM r0, v_att1, BETA, r0
MADM pix_att, v_att2, GAMMA, r0
//

[00043] Finally, FIG. 7 shows a SIMD-ALU structure for the short data processing mode of the method, in accordance with an embodiment of the present invention. As indicated before, this unit is implemented as two scalar ALUs, ALU_0 16 and $ALU_1$18. As shown, both ALU_0 16 and $ALU_1$18 process the aforementioned coefficients, c1, a1h, b0l, a1l, b1l, a0h, b0h, a0l, b1h and c0, with multiplications, shifting and addition operations, The ALU_0 16 and $ALU_1$18 are operative to handle 4 pixels in cycle-shifted operations (each 18x4=72 bits) with the bypass registers 30, 32.

[00044] Thus, the method and interpolation algorithm of the present invention are implemented in floating point arithmetic units with high efficiency and accuracy; a performance improvement is made in triangle interpolation for triangles with size more than 1 pixels compared to prior art methods; and simplification is made of pixel level interpolation hardware by splitting the homogeneous barycentric coordinate calculation to triangle level and pixel level.

[00045] Although the present invention has been described in considerable detail with reference to certain preferred versions thereof, other versions are possible. Therefore, the spirit and scope of the appended claims should not be limited to the description of the preferred versions contained herein.